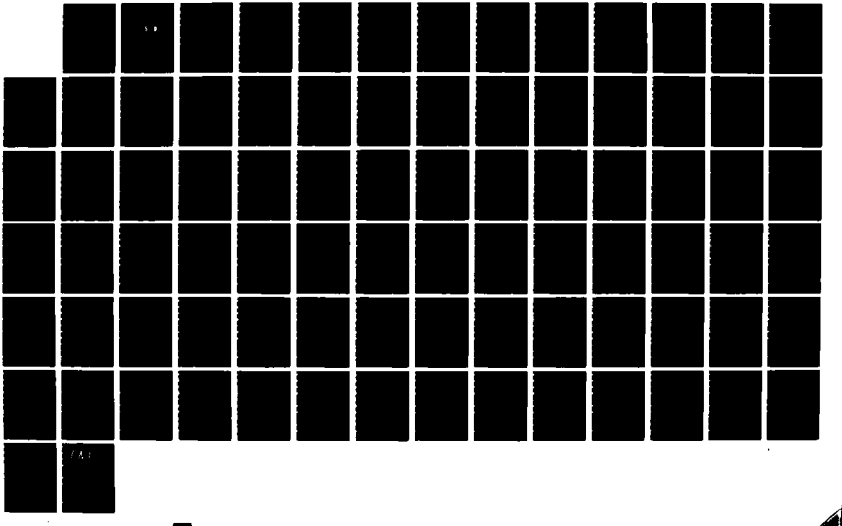
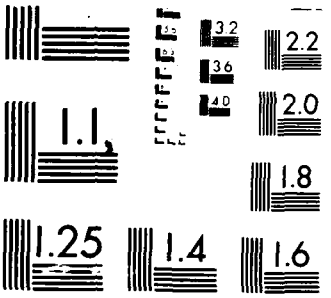
AD-A178 140 IMPROVEMENT OF COMPUTER SOFTWARE QUALITY THROUGH SOFTWARE AUTOMATED TOOLS(U) COMPUTER SOFTWARE ANALYSTS INC INGLEWOOD CA J PEACHER ET AL. 31 AUG 86 1/1

UNCLASSIFIED CSA-85-233-013 F33615-85-C-5109 F/G 9/2 NL





(2)

CSA REPORT 85-233-013

# IMPROVEMENT OF COMPUTER SOFTWARE QUALITY THROUGH SOFTWARE AUTOMATED TOOLS

COMPUTER SOFTWARE ANALYSTS, INC.
333 Centinela Avenue
Inglewood, California 90302





31 August 1986

FINAL REPORT FOR PERIOD COVERING 30 SEPTEMBER 1985 - 30 AUGUST 1986
CONTRACT NO. F33615-85-C-5109 (SBA 98511638)

Prepared for
AIR FORCE BUSINESS RESEARCH MANAGEMENT CENTER
Wright-Patterson AFB, Ohio 45433

CSA REPORT 85-233-013

# IMPROVEMENT OF COMPUTER SOFTWARE

# QUALITY THROUGH SOFTWARE

# AUTOMATED TOOLS

FINAL REPORT

(For Period 30 September 1985 - 30 August 1986)

PREPARED BY

COMPUTER SOFTWARE ANALYSTS, INC.

5217 Wadsworth Road

Dayton, Ohio 45414

## PREFACE

The work documented herein was performed by Computer Software Analysts, Inc. (hereinafter referred to as CSA) under Contract Numbers 98511638 and F33615-85-C-5109. It satisfies the requirements of CDRL Sequence Number 4 (DI-S-3591 A/M, subject Technical Report). Principal investigators for CSA were J. Peacher and P. Ikharebha. CSA acknowledges the outstanding support provided by the Air Force Business Research Management Center (AFBRMC), in particular Captain E. C. Mitchell. In addition, the advice and counsel provided by Mr. Stan L. Brown of the Air Force Contract Management Division (AFCMD) was essential to understanding the needs of the Air Force Plant Representatives office. The cooperation of Westinghouse Corporation (Baltimore) for participating in the demonstration was also appreciated.

TABLE OF CONTENTS

## GLOSSARY OF ACRONYMS AND TECHNICAL TERMS

| | |
|---|---|
| ADF | Automated Design Facility |
| AFB | Air Force Base |
| AFBRMC | Air Force Business Research Management Center |
| AFCMD | Air Force Contracts Management Division |
| AFPRO | Air Force Plant Representative |
| AFR | Air Force Regulation |
| AFSC | Air Force System Command |
| AISIM | Automated Interactive Simulation Modeling System |
| ARTS | Automated Requirements Traceability System |
| ASD | Aeronautical Systems Division |
| CADS | Computer Analysis and Design |
| CCS | Change Control System |
| CSA | Computer Software Analysts, Inc. |
| DAS | Design Analysis System |
| DDPM | Distributed Data Processing Model |
| DEC | Digital Equipment Corporation |
| DFD | Data Flow Diagrams |
| DoD | Department of Defense |
| FAR | Federal Acquisition Regulations |
| FRC | Functional Relationship Charts |
| GFP | Government Furnished Property |
| LCA | Language Control Agent |
| LCF | Language Control Facility |
| IRS | Internal Revenue Service |
| IV&V | Independent Verification & Validation |
| MOA | Memorandum of Agreement |
| QA | Quality Assurance |
| RADC | Rome Air Development Center |
| SBA | Small Business Administration |
| SEI | Software Engineering Institute |
| SOW | Statement of Work |
| SPO | System Program Office |
| STI | Software Tool Information |

GLOSSARY OF ACRONYMS AND TECHNICAL TERMS (Cont'd.)

| | |
|---|---|
| TAR | Test Analysis Report |
| USAF | United States Air Force |
| VHLL | Very High Level Language |
| V&V | Verification and Validation |

LIST OF ILLUSTRATIONS

# EXECUTIVE SUMMARY

## GENERAL

Under contract to the Air Force Business Research Management Center, Computer Software Analysts, Inc. (CSA) was tasked to provide a study that would include the development of a proposed standard that would contain procedures for validating software automated tools. Accordingly, CSA would research, identify and evaluate existing validated tools to understand the current state-of-the-art process. The data obtained from the effort would then form the basis for a standard set of procedures.

Specific study requirements initially required the development and delivery of a detailed Management Plan. This plan was to include AFPROs to be visited and the list of sources to be used as well as CSA's overall approach to the study development. After several reviews, it was completed and delivered.

Initial research identified a number of tools including over 400 from a database located at RADC/COED, Griffiss AFB, NY. Contact was also made with the Software Engineering Institute at Carnegie Mellon University in Pittsburgh to gain from their experience with tools development. Unfortunately since their efforts were primarily ADA development and ADA tools, little was gained from this initiative.

Travel was completed to three contractor facilities: Rockwell (Los Angeles), Boeing (Seattle) and Martin-Marietta (Denver). Discussions with the contractors were primarily concerned with their current and future use of software tools as well as what systems the tools were being applied to. In addition, the contractors were queried as to whether they had developed their own tools or obtained them off-the-shelf. Conversations with the AFPROs had to do with their role in the acquisition process including essentially how they do their job(s). There was also discussion of AFPRO computer resources and anticipated future capabilities. Consensus was that there existed a lack of standardization with software development and documentation. Also the AFPROs identified an acute need for education in the application of tools.

In late December 1985, CSA traveled to AFCMD at Kirtland AFB to provide a status briefing and obtain feedback on CSA's Management Plan. The broadness of the SOW was discussed. In particular, the identification and evaluation of software tools would be limited to those used by DoD aerospace contractors to develop software deliverables to DoD. A list of reference material was provided by AFCMD personnel (i.e FAR, DoD software tools training material and USAF documents) for incorporation as applicable. CSA's basic charter was also to be limited to generating procedures that software developers would use when selecting automated software tools and procedures that AFPROs would use to monitor the contractor's use of tools. This new emphasis on developer use rather than developer validation further refined CSA's future efforts.

As a part of the initial SOW, CSA was also challenged to address whether the Air Force should have a software validation facility or not. This would be a parallel analysis based on insights derived while developing the basic study. The potential need was therefore addressed at every opportunity.

As a result of reviewing information associated with approximately 900 tools, CSA identified 11 of these tools to gain preliminary insight in developing procedures for AFPROs to use. Major areas to be covered within the procedures were established.

In February 1986, CSA attended the Computer Science Conference in Cincinnati, Ohio, sponsored by the Association for Computing Machinery. In addition to attending sessions of fifth generation computing, software engineering and artificial intelligence, private discussions were held with noted individuals from High Technologies Laboratory Research and Carnegie-Mellon University.

As per the contract, a Mid-Term Briefing was presented at AFCMD in March 1986. CSA provided three presentations including one to individuals responsible for three other AFCMD sponsored studies. Further guidance was also given to CSA in that CSA's study was to be heavily slanted for AFPRO usage. More specifically, CSA would develop a standard or checklist to assist the AFPRO in understanding how the developer uses his tool. In

addition, CSA would develop a companion handbook for the AFPRO to clarify the intent and application of the standard. It was envisioned that these two documents would eventually be submitted to the USAF as Appendix 1 and 2 to the overall study. Preliminary discussion also began concerning CSA's demonstration of the above products. Westinghouse Electric Corporation (Baltimore) was suggested as the potential location and tool provider. The USAF was to enlist Westinghouse's participation. In mid April, copies of the Appendix 1 standard were informally provided to the USAF for comments.

In late May, CSA visited Westinghouse and AFSC Headquarters. Westinghouse agreed to participate in the demonstration held at their Baltimore facility on 21 July 1986. Information on the Westinghouse candidate tool was provided to CSA prior to that date. A status briefing was also given to HQ AFSC mission critical computer resources focal points at Andrews AFB. Comments on the Appendix 1 (entitled, "Automated Software Monitoring System") were forwarded to CSA by 6 June 1986. It was also pointed out that the AFSC/PLR had been tasked to develop a number of DoD standards and CSA's product(s) could possibly be used as an initial baseline for one of them.

In addition to the demonstration, CSA delivered the draft study in mid June for review. The final study was due in August.

As to the need for a validation facility, CSA's position presented at the Mid-Term briefing remains essentially the same. That is, the USAF has a role to play in the process but is not to provide a place where contractors will submit and/or run their tools for approval/validation. Rather, the USAF could best provide a library-type facility where information could be retained on file after the tools are validated. An automated data base could easily be structured to maintain the status of the tools and associated documentation. The USAF could assume a semi-active role in the generation of test plans or the approval of test reports. However for the USAF to actually monitor the running of a tool would be very expensive and add very little to the development or validation process.

It would seem appropriate for the USAF to set up workshops or some similar activity to provide this information to the AFPRO personnel. Another suggestion would be computer based education which could be made easily available on site. To develop this methodology and the associated checklists without serious instruction would have questionable utility.

## REFERENCES


a. DoD-STD-2167 - Defense System Software Development

b. STI185 - Software Life Cycle Tools Directory

c. AFPRO 800-14 - Management of Computer Resources in Systems

d. FAR, Federal Acquisition Regulation, (Part 42 Contract Administration)

e. "Characteristics of Software Quality" - Boehm, Brown et al

f. "Tools and Techniques for Structured Analysis and Structures Design" - unpublished BDM document

g. MIL-STD-483 - Configuration Management Practices

h. MIL-STD-490A - Specification Practices

i. MIL-STD-1521B - Technical Reviews and Audits

j. "Computer Science in Focus", Computer Science Conference Proceedings, dated 4-6 February 1986.

# 1.0 PROCEDURE TO EVALUATE SOFTWARE DEVELOPER'S USE OF AUTOMATED TOOLS

## 1.1 Objectives

The objective as described in the contract Statement of Work was for the contractor (CSA) to develop a proposed USAF Standard that contains procedures for validating software automated tools. Over time, however, this was modified somewhat such that the proposed standard would be a document to help the AFPRO better understand how the software developer actually uses his tool. In addition, CSA was tasked to examine the utility of the USAF establishing a tool validation facility.

## 1.2 Scope

CSA was to identify and evaluate existing validated software automated tools to aid in ascertaining the current state-of-the-art process. The data thus derived from this evaluation "will be used to formulate a standard set of procedures for validating software automated tools."

## 1.3 Background

CSA is a California firm which specializes in analysis, simulation, software verification and validation (V&V), and software development. In 1971, the company introduced proprietary software V&V tools to the computer industry enabling customers to provide quality assurance for large software programs. Since that time, these automated tools have been modified for various computers and computer languages. Later, CSA's capabilities were expanded to include systems engineering and computer hardware applications.

As systems engineering performance requirements have become more complex, industry has been forced to convert increasingly from analog to digital technology. Subsequently more dependence has been placed on software reliability to insure end item compliance. Therefore, out of necessity, increased efforts have been expended to develop methodologies to insure the integrity of the software. In modern systems where a single failure may be extremely expensive and/or a safety issue, the importance of

development rigor must be a given. In particular, for real-time systems or for systems performing critical functions (e.g., flight controls), the margin of safety is known to be minimal. This requirement for increased emphasis on software quality assurance has lead to the creation of various methods of verification and validation.

Experience has demonstrated that software of any complexity will contain errors at the completion of development. Manual quality assurance techniques, therefore, evolved to deal with these problems.

As the accelerating need for high-quality software increased, more effective mechanisms were constructed. The result was a vast array of methods, systems, languages and automated tools to assist in the process. Given that the primary role of quality assurance is to provide some degree of confidence that the delivered software product satisfies the user's operational needs, it follows that a series of steps interfaced with the development process is appropriate. In this manner, each phase serves as a benchmark resulting in a verified baseline for the succeeding phase. Essentially, there becomes a one-to-one (parallel) relationship between the development process and the quality assurance program. Unfortunately, there is no single method, tool or technique that can insure accurate, reliable and cost effective software. Therefore, government and industry have expended many manyears of effort creating automated tools and techniques to allow the needed insight into the design, development and validation process for the Programmer Manager, Quality Assurance Engineer/Analyst and the customer. Automated tools are used for the following reasons:

- Early resolution of software errors.
- Operational capability that meets requirements.
- Software life-cycle cost savings.
- Scheduled milestone compliance.
- Increased development control to assist decisions, trade-offs and reporting.
- Trained (in-house) users and maintenance personnel.
- Improved software documentation.
- Audit trail of testing process.

## 1.4 Study Approach

The contract was officially awarded on 30 September 1985. Several minor contractual issues were identified and quickly resolved. Since all dates in the contract were originally predicated on a 3 September 1985 start, everything had to be slipped accordingly to be consistent. Contract monitor agreement was readily obtained. In addition, it was pointed out that contract line 0002AA to deliver software would not be implemented because no software would be developed on this contract. Given the above clarifications, CSA's first task was to develop a Management Plan. A draft was subsequently developed and provided to the USAF for review on 22 October 1985.

During the proposal preparation period, information was collected from a number of software tool developers. This included information from Digital Equipment Corporation (DEC), Reiter Consultants, and Sciences Applications, Inc., as well as the ASD Computer Center. The result was the immediate identification of over 200 tools. In addition to the shear number and variety, it was obvious that categorization was greatly varied. In some cases, tools were classified as to development application (i.e. testing, auditors and traces). In other instances, compilers and other programs integral to the software generation process were classified as tools. It was, therefore, doubtful as to how one would develop a standard approach for the validation of all tools.

In addition to the development of a Management Plan, CSA began the required research to identify industry's automated software tools. A data base was located at RADC at Griffiss AFB. This resulted in obtaining information for more than four hundred tools. Contacts were also made with Carnegie-Mellon University, Software Engineering Institute (SEI), in Pittsburgh, PA. These discussions with R. Ellison revealed that SEI's major emphasis was on ADA development and ADA tools. This information was therefore not included in this study. Tentative travel plans were also discussed with the contract monitor to visit various AFPROs to obtain their views.

During mid-November, two CSA staff members visited three software developers and their associated AFPROs: Rockwell (Los Angeles), Boeing

(Seattle) and Martin-Marietta (Denver). Discussions with the Rockwell AFPRO covered the status of the project, AFPRO problems, AFPRO responsibilities in validating software, and AFPRO's role and relationship with the applicable SPO. CSA was also allowed to observe the B-1B Mock-up and the B-1B Iron Bird. During this visit, the role of quality assurance in the software development process was reviewed. It was pointed out that the AFPRO's role varied considerably depending on the Memorandum of Agreement (MOA) between the AFPRO and the SPO. In particular the SPO has the final decision on all reviews and software acceptance. Rockwell personnel also briefed their software quality assurance procedures in which software tools were an important part.

At Boeing, the initial meeting was limited primarily to software maintenance. Some dynamic analysis tools were mentioned, and most of these tools were developed by Boeing. At a second meeting, Boeing personnel presented CSA with a detailed briefing on Boeing's automated Software Standard (BSWS - 1000) in support of DoD-STD-2167, which establishes the requirements to be applied during the development and acquisition of mission critical computer system software. Also mentioned were 57 software tools that were currently being used. These tools covered various aspects of the software life cycle. An effort is underway to integrate these software tools into packages that cover the entire software life cycle. In turn, these packages will be used throughout the entire Boeing Corporation. In discussions with the Boeing B-1B software test group, how validation testing is performed on application software was covered. This included many Boeing developed analysis tools which are hosted on mainframes. A final meeting with the AFPRO personnel centered on the potential for using automated tools. It appeared that some of the advantages were really not understood. AFPRO education emphasizing the use and application of software appeared warranted.

At Martin-Marietta, two validation techniques were emphasized: rapid prototyping and program design language. As per Boeing, Martin was also developing integrated tool packages that span the software life cycle. At Denver Aerospace (Division of Martin-Marietta), their V&V Plan was briefed. Details were also provided concerning their experience using automated and

manual software validation techniques on a major IV&V project. The AFPRO also provided CSA with a list of requirements for which automated validation tools might be utilized.

In summary, the contractor/AFPRO visits identified the need for unified procedures. It was also agreed that contractor documentation techniques are becoming predominately automated, which emphasizes the need for increased AFPRO training. The establishment of a unified automated software documentation standard to be included in software development contracts would be a great assist.

In late December 1985, CSA traveled to AFCMD at Kirtland AFB. The purpose was to review the information collected to date and to discuss the data obtained from the fact-finding trip to the contractors. Initial subjects covered were the AFCMD role in the acquisition process, AFCMD's automation plans, plus the background leading to CSA's contract. Later discussions included the broadness of the SOW requirements, hardware configuration at AFPROs (and AFCMD), and the effects of MIL-STD-2167 on the AFPROs. It was also pointed out that since software tools (which are used mostly for internal development) are rarely deliverable items, the USAF needs to obtain information as to their selection and use. The final meeting resulted in a further refinement of CSA's task. Basically CSA was instructed to develop procedures for USAF personnel to use when evaluating the software developers selection and utilization of automated software tools.

Based on the contractor visits and the clarification provided by AFCMD, the Management Plan was formally delivered on 17 January 1986. During the same month, information was evaluated associated with almost 900 tools. Eleven were identified as having potential for AFPRO use. This information provided the baseline for "CSA's standards" which are included as Appendices 1 and 2 to this study.

A thorough review of DoD-STD-2167, AFR 800-14 and the FAR (Part 42 Contract Administration) was conducted next. Incorporating these documents with what was learned in the field, CSA was able to develop procedures for AFPRO usage in major areas under investigation. These included:

- Tool description
- Tool name
- Classification
- Abstracts
- Functions
- Parameters
- Validation procedures
- Operating system
- Corporate history
- Tool user's training
- Tool user's supervision
- User's manual

In February, CSA attended the Computer Science Conference in Cincinnati, Ohio which was sponsored by the Association for Computing Machinery. The primary subject areas were fifth generation computing, Software Engineering, Software Specification, and Artificial Intelligence, and Robotics. Of particular interest, the topics under Software Specifications were ENCOMPASS - a tool for the composition of programs and specifications, TIPS - a tool for communicating software requirements, and Case DL - a design tool for specifications and prototyping. Besides the presentations, CSA was able to have side conversations with some of the experts in attendance to obtain their thoughts on tool generation, tool usage, and tool documentation. A conversation with the Director of High Technologies Laboratory Research centered around two types of software tool environments described as "open" and "interactive". The former environment pertains to off-the-shelf software whereas the interactive environment tools are those that have been incorporated into the development package. A session with the Carnegie-Mellon University Computer Science Department Head, was also informative. This resulted in the identification of two more expert sources for CSA to contact. CSA also talked with various publishers and vendors concerning printed material on software tool evaluation. No one appeared knowledgeable of publications on the subject.

Work started in preparation for the Mid-Term Briefing at AFCMD. In addition to providing status information, CSA was striving to address its basic charter to develop a draft standard and address the need for a validation facility. After all the research and discussions, five evaluation categories appeared to be appropriate for inclusion in the standard:

a. Software tool identification
b Software tool schedule
c. Tool utilization and training
d. Validation of software tool and evaluation of tool documentation
e. Configuration change control

The format of the report could now be better defined. CSA's approach has been described in Section 1.0 and the need for a validation facility in Section 2.0. Appendix 1 (Automated Software Tool Monitoring System) is what the original SOW described as AF standard with Appendix 2 being a handbook written to assist the AFPRO; Appendix 3 is a checklist for applying Appendices 1 & 2. Appendix 4 is an overview of Tool's Characteristics. Prior directions to potentially include lists of useful tools and matrices of validation processes were no longer to be addressed.

The Mid-Term Briefing(s) were conducted in late March. CSA was one of four contractors presenting information on various aspects of software procurement. For instance, one briefing was concerned with the cost of software development, another with contract administration and one addressed quality assurance.

CSA actually participated in three different forums. The first was a presentation to the AFCMD staff and to the presenters of the other studies. The second was a brief presentation to Col. Roelig and his staff. The final presentation was for the AFPRO MCCR focal points. Basic agreement was reached on how CSA was to proceed with the study. It was also determined that the USAF would approach Westinghouse (Baltimore) for participation in the demonstration. Conceivably, Westinghouse would be the candidate tool developer and provide a tool to be subjected to CSA's procedure. CSA's presentation also outlined what a viable USAF software validation facility should entail. If one can accept the idea that a

validation facility is an activity that determines the correctness of a program, or software project in terms of its functional requirements, the issue can be appropriately discussed. As covered in detail in Section 2, CSA envisioned this capability to fulfill a library/approval role rather than a place for physical demonstrations.

In April 1986, a draft of CSA's Appendix 1 (i.e. "Standard") was delivered to the Air Force for comment. Feedback was all positive. The Air Force also decided that the final presentation would be at AFCMD during the week of 11 August 1986.

CSA was then asked to provide a list of information that the USAF would need to provide for the demonstration. This later required some clarification since the information would actually be generated by Westinghouse for USAF use. In preparation for this, CSA traveled to Westinghouse with an outline of the anticipated demonstration. The subsequent discussions were very positive resulting in Westinghouse nominating one of its tools and the use of its facility for the demonstration. In addition, Westinghouse agreed to provide CSA with information on the subject prior to the event taking place. After these discussions, CSA provided a status briefing to AFSC Headquarters personnel at Andrews AFB. The possibility of using CSA Appendix 1 as a foundation for a DoD standard was discussed.

In summary, the schedule for contract completion included delivery of a draft study on 16 June 1986, a demonstration conducted at Westinghouse, and a final briefing at AFCMD. The final study with USAF comments incorporated would be delivered in August 1986, the scheduled final month of the contract.

## 2.0 ANSWER TO THE QUESTION "DOES THE AIR FORCE NEED A TOOL VALIDATION FACILITY?"

### 2.1 Validation

Historically, software has required extensive operational usage before a respectable level of confidence is achieved. Subsequently, the software community (including DoD developers) have been expanding development and testing techniques to include the use of automated auditors, analyzers and various test tools. This new automated QA methodology has increased the effectiveness and reliability of software while it has reduced the time required to "finalize" the system. The end product is typically more reliable resulting in a higher degree of confidence.

The rapid advance in system sophistication during the last few years has given rise to an increased requirement for more powerful and efficient software systems which will replace those already proven and in the inventory. This, in turn, has lead to an acceleration of the need for validated tools for these new applications. Subsequently higher confidence, especially in military systems, must be achieved in a shorter period of time (preferably prior to final acceptance). By providing a set of standards and procedures for developers of QA tools, these improvements should be realized. It seems only reasonable that by instilling the same discipline in the generation of automated tools as the other portions contributing to the weapon system, the final product will be of higher quality.

### 2.2 Tool Availability

As would be expected with the growth of the software industry, there has been a similar increase in automated tools. In addition to CSA, many firms have been involved, and as can be seen, for many different reasons and applications.

Caine, Farber & Gordon Inc. developed the S-FORTRAN language. It is an extension of the FORTRAN language to allow easy, efficient and reliable programming in a FORTRAN environment. The language results from the adjunction of a carefully chosen set of control structures. These extensions encompass all those that have been proposed in the literature and proven to be useful.

The University of Michigan produced the Problem Statement Language (PSL) and the Problem Statement Analyzer (PSA) as computer-aided techniques for structured documentation and analysis of information processing systems. They are tools for describing information processing systems, recording such descriptions in machine-processible form, and storing in a database. Specifically, PSL is a language for describing systems; it is not a programming language. PSA is a software package that is used to check data as it is entered into the computer, store it and produce reports.

The Digital Equipment Corporation developed VAX-11 DEC/CMS which is a program librarian for software development and evolution on a VAX/VMS operating system. It is a set of commands that enable software developers to manage the files of an ongoing project.

Reifer Consultants, Inc. list three types of automated tools available from various sources. They are: Simulation Packages, Requirement Packages and Configuration Management Packages. Several examples of each are:

- Automated Interactive Simulation Modeling System (AISIM).
- Computer Analysis and Design System (CADS).
- Distributed Data Processing Model (DDPM).
- Automated Design Facility (ADF).
- Automated Requirements Traceability System (ARTS).
- Design Analysis System (DAS).
- Batch Librarian (B-LIBEXEC/7).
- Change Control System (CCS).
- Common Block Generation Program (COMGEN).

Science Applications Inc. has developed Science Applications Inc. Software Design and Documentation Language (SAI-SDDL). It is a licensed computer program which aids in the documentation, as well as the design of, computer software. The three components are the processor (a PASCAL program), the language and the methodologies. It can operate directly on FORTRAN, PASCAL or SIMSCRIPT source code.

Historian Plus was produced by Opcode, Inc. It is a software package that allows the user to store and update an entire library of source materials, while keeping track of all changes. It executes in both batch

and interactive modes. At one time it was capable of running on eleven (11) mainframes and nineteen (19) different operating systems. Some of its users include several Japanese clients.

Logicon's Automated Requirements Engineering (LARE) and Computer Response Time Simulation (CRTSIM) have been developed to aid the system engineering process. The characteristics of quality requirements are defined as testable, unambiguous, consistent, traceable, complete and maintainable. It also assists configuration management and other management functions. Logicon's Strategic Mission Data Preparation System (SMDPS) is an interactive mission planning system that produces mission data on magnetic tape for strategic avionics systems. It employs a complex database management system to manipulate a large amount of mission-related data.

ARINC Research Corporation System Testability and Maintenance Program (STAM) is a computer-aided testability and fault diagnosis system. A dependency analysis algorithm identifies all higher-order dependencies and their implications with respect to fault isolation and maintenance. It also provides testability measures that are useful in comparing competing designs and evaluating the effects of testability design changes (e.g., test-point relocations).

The Aeronautical Systems Division Language Control Facility Newsletter (April 1985) lists J73 and 1750/1750A software. This includes 1750/1750A tools, tools for other embedded computer targets, support tools and compilers. Of the fifty (50) tools listed, approximately fifteen (15) different contractors are listed as the developers.

During the course of CSA's research activities, over 900 tools (as described by various developers) were identified. The magnitude of the number of different languages and different types of hardware was obviously very large. In addition, with the seemingly constantly changing state-of-the-art in the computer business, these numbers will continue to rapidly increase.

### 2.3 ASD Language Control Facility

The Language Control Facility (LCF) is operated by the Language Control Branch at the ASD Computer Center at Wright-Patterson AFB, Ohio.

It is supported by both government and contractor personnel. Its basic charter is to support the Language Control Agent (LCA) in implementing JOVIAL J73 as a standard Air Force language for avionics embedded computer applications. Air Force Regulation 300-10 requires all JOVIAL compilers (intended for use in Air Force avionics embedded computer applications) to be validated before being accepted for Air Force use. MIL-STD-1589C (USAF), entitled JOVIAL (J73), provides the basis for the validation process.

As a service, the LCF conducts both formal and informal compiler validations. For a formal validation, a Memorandum of Agreement is established between the LCA, LCF and the requesting Program Office. Then, based on inputs from the compiler developer, the LCF prepares a test plan. The LCF next performs a prevalidation site inspection (as required), oversees and controls execution of the JOVIAL Compiler Validation System at the site specified by the Program Office. Upon completion, the LCF analyzes the results and prepares a full Test Analysis Report (TAR). The TAR includes the identification of all failed tests, as well as other inconsistencies, with the MIL-STD. For informal validations, the LCF provides the requester with a skeleton test plan and assists with the procedures for loading and using the JOVIAL Compiler Validation System. After execution, the requester mails the results to the LCF for analysis which results in the preparation of an informal TAR.

## 2.4 Air Force Requirement

As was discussed at CSA's Mid-Term presentation, the Air Force needs to be involved with automated tool development as well as tool usage. The pertinent question is to what degree, based on the magnitude of the problem. If a validation facility can be defined as an organization that determines the correctness of a program or software product in terms of its advertised functional requirements, then it might be a realistic undertaking.

One could envision a large building filled with many machines and staffed with an appropriately large number of systems analysts and programmers, etc. Then one could envision individual software developers bringing their tapes and card decks and so forth to be run and therefore

validated. Even if it were possible to obtain all the required mainframe, peripherals and support equipment, it would soon be out of date. So, in addition to the large initial investment there would be a continuous update expense.

A more practical solution could be described as a "Government Furnished Property (GFP) Facility" in which a software tool database was maintained. It could be interactive as per the ASD facility or simply provide a library function. In either case, the facility would definitely contribute to adding discipline to the process. Responsibilities could include:

Managing software tools
Managing tool documentation
Tracking tool development
Overseeing validation of candidate tools
Maintaining qualified tools list

The Data and Analysis Center for Software (DACS) which is already a central source for usable data and software technology could be utilized. It is currently developing and maintaining a database and providing rapid response to technical inquiries.

## 3.0 Overall Summary

### 3.1 Conclusions

A software automated tool is a software program and should most appropriately be treated as such. During development, therefore, it should be subjected to the same rigor as delivered software, (i.e., testing, documentation, etc.). In addition, guidelines are necessary to avoid having the USAF in the middle when there is a debate between the software program developer and the software tool developer, particularly if the two are procured separately. A much needed by-product of this structured approach will be a better definition for tools. With the recent proliferation of programs labeled as tools (for testing, IV&V, compiling and a multitude of other uses), categorization is obviously needed. As an aid in

this process, the USAF could provide the regulatory function by providing the standards and approval process. However, recognizing the large number of tools which are constantly being upgraded, it would definitely be impractical for the USAF to require a physical demonstration of each in a government facility.

## 3.2 Recommendations

Appendix 1 is a document in military standard format which will assist the AFPRO in the development/procurement process. It will greatly assist the AFPRO (and in turn the USAF) to understand how a particular tool was developed as well as how and why it was used. This document would be a great help to both the developer and the USAF in the software development process. Appendix 2 is a handbook which will assist the AFPRO with the implementation of Appendix 1.

The USAF needs an information data base on software tools. The function of this software tool database facility would be to hold/disseminate software tool information to industry and government organizations. There would be minimal cost to the government since some facilities for deliverable software tools already exist at RADC and WPAFB. The utilization of this facility would aim towards the reduction of software tool duplication and waste. It would include a provision for a move in the direction of standardization.

A Computer Based Education (CBE) software package is also recommended for those AFPROs with lesser knowledge of software tools. No such package is available off-the-shelf but one could be readily developed under contract. The AFPRO would also benefit from workshops to better understand the application of Appendix 1 and Appendix 2 for the evaluation of software tool procedures.

# AUTOMATED SOFTWARE TOOL MONITORING SYSTEM

(This is a proposed Standard to be used by the USAF to better understand the use of software tools in the development of the software life cycle. Its application to non-delivered software will provide the greatest benefit.)

APPENDIX 1

## 1.0 SCOPE.

1.1 Applicability. This document shall apply to all automated software tools (deliverable or non-deliverable) used by contractors or subcontractors in the development of embedded (application) software systems. In this document the term automated software tool (or software tool) will be defined as, "Firmware used to support in the development and maintenance phases (i.e. requirement analysis, design, coding, testing and configuration management) of a software system life cycle.

1.2 Contractual Intent. This document requires the establishment and implementation of an automated software tool (AST) monitoring program by the contractor or subcontractor. The objective of this document is to assure that AST is identified, documented, tested, validated and maintained in a manner that will provide AST information to other users. Also a move in the direction of AST requirement standardization and control of AST duplication.

1.3 Relation to Other Contractor Requirements. This document and any procedure or document executed in implementation thereof, shall be in addition to other contract requirements. The monitoring program requirements set forth in this document shall be satisfied in addition to all detail requirements contained in the Statement of Work or in other parts of the contract.

The contractor is responsible for compliance with all provisions of the contract and for furnishing specified supplies and services which comply with all requirements of the contract. If any inconsistency exists between the contract schedule or its general provisions and this document, the contract schedule and the general provisions shall control.

## 2.0 APPLICABLE DOCUMENTS:

2.1 Amendments and Revisions. Whenever this document is amended or revised subsequent to its contractually effective date, the contractor may follow, or authorize his subcontractor to follow, the amended or revised document, provided no impact on schedule or increase in cost, price, or fee is required. The contractor shall not be required to follow the amended or revised document except as a formally authorized modification to the contract. If the contractor elects to follow the amended or revised document, he shall notify the contracting officer in writing of this election. When the contractor elects to follow the provisions of an amendment or revision, he must follow them in full.

2.2 Ordering Government Documents. Copies of specifications, standards, and documentation required by contractors in connection with specific procurements may be obtained from the procuring agency, or as otherwise directed by the contracting officer.

3.0 GENERAL REQUIREMENTS.

3.1 The contractor shall implement and document an AST monitoring program. The contractor shall maintain information concerning this program in the following format:

a. Volume 1 - General Information

b. Volume 2 - Specific Software Tool Information

3.1.1 Volume 1 - General Information. This volume will consist of five sections that present generic information concerning the ASTs used on a project.

3.1.1.1 The first section is the Introduction. The Introduction shall contain:

a. Identification of the project(s) to which the AST has or will be applied.

b. A listing of all other software tools (in software life cycle order) that interface with the software tools.

c. A brief functional description of the AST.

3.1.1.2 The second section is a Master Schedule that indicates the date all software tools (listed above) will become operational. A more detailed schedule for each software tool will be presented in Volume 2.

3.1.1.3 Third is the Configuration Change and Control Plan section. In this section the contractor shall describe how control of the software tools' configuration is maintained.

3.1.1.4 The Personnel Training Plan section is the fourth section. The contractor shall describe his overall and individual software tool training plan. Also, a training schedule tracking each employee's progress throughout the training process shall be provided.

3.1.1.5 Section five, Other Information, will contain any other generic information (not discussed above) applicable to the ASTs.

3.1.2 Volume 2 - Specific Software Tool Information. This volume shall describe information pertaining to each software tool used in the project. Volume 2 contains specific software tool information. Each software tool will be described by five (5) categories of information as follows:

a. Software Tool Identification
b. Software Tool Schedule
c. Validation of Software Tool and Evaluation of Documentation
d. Configuration Change and Control Status
d. Other Information

3.1.2.1 Software Tool Identification. When identifying the software tool, the following will be addressed:

Software tool taxonomy (General Characteristics - See attachment for each tool)

3.1.2.2 Software Tool Schedule. A detailed schedule outlining the various activities necessary to make the software tool operational is required. Activities may vary from a straight acquisition and installation of a software tool to a full-scale software tool development effort.

3.1.2.3 Validation of Software Tools and Evaluation of Documentation. The elements of this category describe the methodologies and techniques used to validate the software tool and evaluation of the software tool documentation. The purpose of the test is to determine if the tool does what it claims, and if the documentation reflects the software tool that was tested. The following documentation procedures shall be required:

a. Analysis of software requirements to determine testability
b. Review of test requirements and criteria for adequacy
c. Verification that tests are conducted in accordance with approved test plans and procedures.
d. Review the documentation to verify that it corresponds to the actual test results.
e. The contractor shall ensure that test related media and documentation are maintained.

3.1.2.4 Configuration Change and Control of Software Tool and Documentation. This category is concerned with the management of any changes to the baseline software tool. Subcategories to be addressed include:

a. Software tool baseline date
b. Software tool baseline configuration (if different from software tool identified above)
c. Track of changes to software tool baseline
d. Software tool documentation baseline date
e. Description of baseline documentation
f. Track of changes to baseline documentation

3.1.2.5 Other Information. This section will be reserved for information not covered in the above section applicable to the AST.

ATTACHMENT 1-1

SOFTWARE TOOL TAXONOMY

The information provided in this attachment (which is part of the AST monitoring system) will enable the AF to update its software tool database at Griffis AFB, NY, and provide minimum records for the non-deliverable software tools.

THIS FORMAT IS IN COMPLIANCE WITH THE "SOFTWARE LIFE CYCLE TOOLS DIRECTORY" DATABASE. THE CONTRACTOR SHALL IDENTIFY THE TOOL AS FOLLOWS:

TITLE: The name of the tool in its long form.

DATE TOOL INFORMATION UPDATED: The date the tool information or updated tools information was provided.

DATE OF DEVELOPMENT: The date the tool development was completed (month/day/year).

CLASSIFICATION: One or more of the following classes that best describe the tool's application:

- Software Management, Control, and Maintenance Tools
- Software Modeling and Simulation Tools
- Requirements and/or Design Specification and Analysis Tools
- Source Programming Analysis and Testing Tools
- Program Construction and Generation Tool
- Software Support System/Programming Environment Tools
- Other

FEATURES: The input, function, and output of a tool includes the following data elements:

SUBJECT: The input which is subjected to the main functions performed by a tool.

CONTROL INPUT: The types of operations and details associated with the operations.

TRANSFORMATION: Changes that take place on the input to a tool while it is being processed.

STATIC ANALYSIS: Operations on the subject without regard to its executability.

DYNAMIC ANALYSIS: Operations that are determined during or after execution takes place.

MACHINE OUTPUT: Output that can be directed to a target machine or to another tool for further processing.

USER OUTPUT: The types of information that are returned from the tools to the human user, and the forms in which these outputs are presented.

STAGE OF DEVELOPMENT: What development stage the tool is in: Concept, Design or Implemented.

PURPOSE OF DEVELOPMENT: The purpose for which the tool was developed, can be either Research, Experimental, or Production.

TOOL PORTABLE: Whether or not the tool can be easily transferred from one machine to another without extensive modification.

PRICE: The purchase/lease price of the tool.

HOST COMPUTER: The hardware manufacturer and identification of the machine on which the tool was developed.

TARGET COMPUTER(S): The hardware manufacturer, identification of the machine(s) necessary for the operation of the tool.

WORDSIZE: The wordsize of the target computer(s).

OPERATING SYSTEM: The operating system necessary for the use of the tool.

OTHER SOFTWARE/UTILITIES REQUIRED: Other software necessary for the use of the tool.

TOOLSIZE:

SOURCE WORDS: The number of bytes of the tool's source code.
OBJECT WORDS: The number of bytes of the tool's object code.
SOURCE LINE OF CODE: The number of source code lines of the tool.
MEMORY REQUIREMENTS: The number of bytes of memory required for the operation of the tool.

IMPLEMENTATION LANGUAGE: The language(s) in which the tool is written.

TOOL AVAILABLE: Whether or not the tool is available to other users (Yes/No).

PUBLIC DOMAIN: Whether or not the tool is in the public domain (Yes/No).

TOOL SUPPORTED: Whether or not the tool is supported (Yes/No).

TOOL SUPPORTER: The organization responsible for maintenance and/or configuration control.

ABSTRACTIONS (COPYRIGHTS, LICENSES, GOVERNMENT APPROVAL, ETC.): The restrictions on the availability of the tool.

TOOL SUMMARY: A brief paragraph clarifying the features the tool provides; indicating, if possible, the number of users; and discussing the experiences with the use of the tool such as performance, applications, or any other pertinent information.

DOCUMENTATION: The written documentation, documentation sources and document reference numbers.

REFERENCES: Articles or publications that discuss the tool and experiences with the tool.

CONTACT FOR MORE INFO: The contact to obtain more information about the tool (name, organization, address, phone).

DEVELOPER: The tool developer (name, organization, address, phone).

DISTRIBUTOR: The tool distributor (name, organization, address, phone).

SPONSOR: The agency who sponsored the development of the tool (name, organization, address, phone).

# HANDBOOK FOR AUTOMATED SOFTWARE TOOL

# MONITORING SYSTEM

APPENDIX 2

## INTRODUCTION

This document and Automated Software Tool Monitoring Program (Appendix 1) are based on established Department of Defense (DoD) concepts and policies which provide that:

a. Contractors are solely responsible for the control of product quality and for offering to the Government for acceptance only products determined by them to conform to contractual requirements.

b. Government representatives are responsible for determining that contractual requirements have, in fact, been complied with prior to acceptance of the product.

c. Final decision of product acceptability is solely the responsibility of the Government.

The contractor, in accordance with Appendix 1, must design and maintain an effective and economical monitoring program that includes both processes and products which makes data available to the Government adequate for use in establishing AST acceptance criteria. Facilities, products, and management techniques vary so widely within the broad pattern of national security industrial establishments that this evaluation handbook cannot provide detailed information to cover all intentions. Instead, it reflects the most reliable quality program control patterns used by much of American industry. It encourages the training of planners and evaluators in all areas that affect the program. The emphasis throughout this handbook is on the planning and execution of a comprehensive monitoring program. The evaluation of such a program depends upon how well decision criteria have been selected, applied and enforced.

A consistent format has been followed throughout this handbook. In order to relate the program evaluation suggestions (directly as possible) with

the requirements of Appendix 1, each subsection of Appendix 1 is quoted verbatim and is followed by appropriate comments as follows:

SUBSECTION OF APPENDIX 1

A. "Review of Requirements" - Discussion of the requirements set forth in the subsection.

B. "Application" - Description and examples of practices applied by contractors in the past that are typical and illustrative

C. "Criteria for Evaluation" - Questions which should be asked to evaluate that particular part of a contractor's quality program.

It is most important to note that the questions contained in the various "Criteria for Evaluation" are essentially YES/NO questions. Asking and answering them alone will not provide a thorough and complete evaluation of a contractor's monitoring program. The questions serve only as indicators and reminders of important points to cover; the evaluation is expected to cover them in appropriate depth and detail to assure an effective and complete evaluation.

EVALUATION OF A CONTRACTOR'S SOFTWARE TOOL PROGRAM

1.0 SCOPE.

1.1 Applicability. This document shall apply to all automated software tools (deliverable or non-deliverable) used by contractors or subcontractors in the development of embedded (application) software systems. In this document the term automated software tools (or software tools) will be defined as, "Firmware used to support in the development and maintenance phases (i.e., requirements analysis, design, coding, testing, and configuration management) of a software system life cycle.

1.2 Contractual Intent. This document requires the establishment and implementation of an AST monitoring program by the contractor or subcontractor. The objective of this document is to provide for the user (AFPRO) a better understanding of how contractors are using software tools, and a method of evaluating the software tools.

1.3 Relation to Other Contract Requirements. This document and any procedure or document executed in implementation thereof, shall be in addition to other contract requirements. The monitoring program requirements set forth in this document shall be satisfied in addition to all detail requirements contained in the Statement of Work or in other parts of the contract.

2.0 APPLICABLE DOCUMENTS:

2.1 Amendments and Revisions. Whenever this document is amended or revised subsequent to its contractually effective date, the contractor may follow, or authorize his subcontractor to follow, the amended or revised document, provided no impact on schedule or increase in cost, price, or fee is required. The contractor shall not be required to follow the amended or revised document except as a formally authorized modification to the contract. If the contractor elects to follow the amended or revised document, he shall notify the contracting officer in writing of this election. When the contractor elects to follow the provisions of an amendment or revision, he must follow them in full.

2.2 Ordering Government Documents. Copies of specifications, standards, and documentation required by contractors in connection with specific procurements may be obtained from the procuring agency, or as otherwise directed by the contracting officer.

3.0 GENERAL REQUIREMENTS:

3.1 The contractor shall implement an AST monitoring program in the following format:

a. Volume 1 - General Information

b. Volume 2 - Specific Software Tool Information

3.1.1 Volume 1 - General Information. This volume will consist of five sections that present generic information concerning the AST used on a project.

3.1.1.1 Introduction. This is an identification of the Software System and the project(s).

A. REVIEW OF REQUIREMENTS.

Appendix 1, para. 3.1.1.1, requires the contractor to: a) identify the project(s) to which the AST has been or will be applied; b) list all other software tools (in software life cycle) that interface with the AST; and c) give a brief functional description of the AST.

B. APPLICATION.

During the development cycle of software, many individual functions are performed to ensure quality of the software. This includes reviews of software documentation from the initial specification documents through the final test reports. Many software tools may have been used to support these individual functions. The contractor governed by the above requirement will identify the project and provide the listing of all software tools that interface with this software. To better understand the specific function of the AST, the contractor will provide a brief functional description of the input and output of the AST. See Figure 2-1 of AST General Information.

C. CRITERIA FOR EVALUATION.

- Has the contractor identified the project(s) to which the AST has or will be applied?
- Are software tools listed in software life cycle?
- Has the AST been evaluated under DoD-STD-2167?
- Is all documentation available to government personnel, and furnished upon request?
- Has the contractor provided the AST functional description?
- Does the description provide information on input, output and expected results?

| IDENTIFY PROJECT | LISTING OF AST | AST FUNCTIONAL DESCRIPTION | (Software Life Cycle Order) |
|---|---|---|---|
| F 16 | 1) TRANSFOR | TRANSFOR is a pre-processor that extends the FORTRAN language for Structured Programming. It allows the programmer to write efficient, structured code-free "go to" statements. The use of statement numbers is eliminated and the nesting of control structures enforced. TRANSFOR and FORTRAN intermixed. The resultant software is more readable because of the automatic identification of control statements by which nesting is depicted. Sneak paths and unanticipated execution sequences are rare, since the programmer is able to see primarily straight forward logic. | Requirements/Design Specification Analysis |
| | 2) DAVE | DAVE performs data flow analysis of FORTRAN programs in order to detect variable usage anomalies. The system provides good documentation, reliability, ease of use fair cost effectives, and user support. DAVE detects data usage anomalies such as: references to undefined variables, unreferenced variable definitions, uninitialized variables (local or common), and unused variables. | Source Program Analysis & Testing |
| | 3) CAT | These tools are used to assure the validity of the computer program configuration integrity by selective tracing changes to the configuration of the computer program. | Software Management, Control and Maintenance |



FIGURE 2-1

3.1.1.2 MASTER SCHEDULE PLAN. This is primarily a listing of all tools and major milestones. The plan shall reference or document the starting dates of all ASTs based on the date the ASTs become operational.

A. REVIEW OF REQUIREMENTS. Appendix 1 of para 3.1.1.2 requires contractors to establish a master schedule indicating the date the software tool will become operational.

B. APPLICATION.

Contractors usually have formal procedures for describing the master schedule on a given project. This procedure's formality will depend on a contractor and the contract under consideration. The real significance is whether the procedures provide adequate information. The AFPRO will use his judgement in exercising this requirement as optional, particularly since developed software tools may not be applicable.

C. CRITERIA FOR EVALUATION.

- Has the contractor provided the master schedule plan?
- Are there provisions for monitoring and tracking the changes that may have an effect on the plan?
- Are the listings of ASTs and the major milestone schedule provided?
- Has the contractor shown a close out of completed tasks?

3.1.1.3 Configuration Change and Control Plan. The contractor shall be required to produce a plan for all contract and modification procedures made to the AST.

A. REVIEW OF REQUIREMENTS.

Appendix 1 of para. 3.1.1.3 requires the contractors to describe how control of the AST's configuration is maintained.

B. APPLICATION.

Once a baseline has been established for AST or supporting documentation, i.e., specification, design, test plan, etc., the integrity of the baseline or documentation is protected to ensure that there are not any unauthorized changes. It is important

that the AST change control plan identify the authority to enter material under configuration control and identify the authority for removal of controlled items from the configuration management activity. The plan shall provide explicit instructions for identification of baseline materials and subsequent revisions or versions. The plan shall provide procedures that will preclude the control facilities from being used as a repository for unapproved, or uncontrolled ASTs.

C. CRITERIA FOR EVALUATION.

- Has the Contractor provided procedures to ensure that changes to the baseline specification and documentation are authorized?
- Is the Contractor complying with internal procedures for placement and removal of items from the control facility?
- Does the Contractor's plan preclude the control facilities from being used as a repository for unapproved or uncontrolled ASTs?
- Has the contractor followed procedures to identify baseline items and subsequent revision or version of the AST?

3.1.1.4 PERSONNEL TRAINING PLAN. The plan shall reference or document procedures for the training plan on AST.

A. REVIEW OF REQUIREMENTS.

Appendix 1 of para 3.1.1.4 requires contractors to describe the overall and individual software tools training plan. This also includes training schedule tracking each employee's progress throughout the training process.

B. APPLICATION.

The application of AST at any given phase of the development is very important. To ensure the quality of software, it is neces-

sary that individuals have proper knowledge of operating AST. Before such individual operates a tool, it is necessary to establish training plans. These plans shall describe the overall training of AST and the individual training plans. Appendix . also requires the contractor to provide a schedule that would track/monitor this training plan.

C. CRITERIA FOR EVALUATION.

- Has the contractor provided a training plan? Who monitors this training plan?
- Has a procedure been developed to track the training system
- How well are these training plans administered?

Note that the training plans for all ASTs will be applied to the individual AST. Hence individual AST training is not discussed.

3.1.1.5 Other Information.

This section will be reserved for other information that was not covered in the above sections. Typically it will cover generic information that is also applicable to ASTs as specified in Appendix 1 of para. 3.1.2.4.

3.1.2 Volume 2 - Specific Software Tool Information. This volume shall describe information pertaining to the AST used in the project. The AST will be described by five (5) categories of information as follows:

a. Software Tool Identification
b. Software Tool Schedule
c. Validation of Software Tool and Evaluation of Documentation
d. Configuration Change Control Status
e. Other Information

3.1.2.1 AST Identification. The contractor shall provide a description of the AST and the important specifications. This information will enable the Air Force to update its software tool database at Griffiss AFB, NY, and

provide minimum records for the non-deliverable software tools.

A. REVIEW OF REQUIREMENTS.

Appendix 1 of para. 3.1.2.1 requires that when identifying the AST, the following will be addressed:
Software tool Taxonomy (General characteristics - see Atch for each tool of Appendix 1)

B. APPLICATION.

Examples and information needed to help identify the AST are: title of the AST, date the AST information was updated, date the AST was developed, classification of the AST (i.e., what phase of software development are the ASTs applicable). If such information is not relevant to the AST, the contractor shall respond "not applicable". However, if such information is applicable to the AST, but the information was not given, the AFPRO shall then take necessary action to obtain such information.

C. CRITERIA FOR EVALUATION.

- Has the contractor provided information to properly identify the AST?
- For the information not provided, are there special means to furnish the government the information missing?
- Is the AST information accurate and complete?
- Is the AST approved to be used on this project?
- Is the AST in development stage?
- If yes, is the development included in the project contract?
- Has the contractor researched current government tools to avoid duplication?
- What are the advantages of this AST compared to the existing software tools?
- Is the AST a non-deliverable item?
- If yes, are there effective means for providing information to government personnel during and after the development of software life cycle?

3.1.2.2 Software Tool Schedule. Appendix 1 shall require contractors to provide all information regarding the AST milestone schedule.

A. REVIEW OF REQUIREMENTS.

Appendix 1 of para 3.1.2.2 requires contractors to provide a detailed schedule outlining the various activities necessary to make the software tool operational. Activities may vary from a straight acquisition and installation to a full-scale software tool development effort.

B. APPLICATION.

During the development of software, contractors shall identify the detailed status of the AST. Contractors may already have in their library an AST to be used on a particular task, and often the AST may need some modifications to meet the need of that particular task. In the case where an AST has not been developed or needs modification, the program requires contractors to provide in detail, the status at any given point of the AST and when the AST becomes operational.

C. CRITERIA FOR EVALUATION.

- Has the contractor provided a milestone schedule for the AST?
- Is information provided in the milestone schedule accurate?
- Has the contractor provided date(s) the AST will become operational?

3.1.2.3 Validation of Software Tools and Evaluation of Documentation. The elements of this category describe the methodologies and techniques used to test the software tool and evaluation of the AST documentation. The purpose of the test is to determine if the tool does what it claims, and if the documentation reflects the software tool that was tested.

A. REVIEW OF REQUIREMENTS.

Appendix 1 of para 3.1.2.3 requires documentation procedures for the following:

a. Analysis of software requirements to determine testability
b. Review of test requirements and criteria for adequacy
c. Verification that tests are conducted in accordance with approved test plans and procedures
d. Review the documentation to verify that it corresponds to the actual test results.
e. The contractor shall ensure that test related media and documentation are maintained.

B. APPLICATION. The qualification of software can be accomplished through the application of stringent testing. Each phase of the development of a software system will normally require testing and validation prior to continuing to the next step. For example, some computer programs are tested prior to integration or subsystem testing. If modules are produced in a top down order, top down testing will be employed. However, the contractor shall be required to identify those procedures or techniques to ensure the AST has been tested in accordance with its test requirements and specifications. Test procedures shall include prevention, detection, diagnosis, recovery and correction of errors.

C. CRITERIA FOR EVALUATION.

- Has the contractor identified test activities?
- Has the contractor provided test procedures and documentation for internal testing and evaluation?
- Have various levels of testing been identified and scheduled as required?
- Do test procedures comply with the test specification, data item descriptions and other contractual requirements?
- Are test results actual findings of the test?
- Are test-related media and documentation maintained to allow repeatability of tests?
- Are software tool requirements testable?

- Are all software and hardware used to develop the AST acceptable to the government?
- If acceptable, is there necessary documentation provided to substantiate this fact?

3.1.2.4 Configuration Change and Control of Software Tool and Documentation. This category is concerned with the management of any changes to the baseline software tool.

A. REVIEW OF REQUIREMENTS.

Appendix 1 of para. 3.1.2.4 requires the following sub-categories to be included:

1. AST baseline date
2. AST baseline configuration (if different from software tool identification above)
3. Track of changes to software tool baseline
4. Description of baseline documentation, date(s) of changes, and track of changes to baseline documentation.

B. APPLICATION.

Once a baseline has been established for an AST, the integrity of the baseline and documentation is protected to ensure that there are no unauthorized changes. However, if changes are made to the baseline, the contractor shall provide date(s) the changes were made, types of changes, and the overall effect the changes will have on the AST. Contractors shall provide tracking procedures for all changes to the baseline. The AST documentation shall reflect all changes and the tracking of baseline documentation.

C. CRITERIA FOR EVALUATION.

- Has the contractor established the baseline date?
- Have changes been made to the AST baseline?
- Has the contractor provided description for changes made to AST documentation.
- Have dates been provided to baseline changes?

- Has the contractor provided AST baseline configuration?
- Has the contractor provided procedures to track changes made to the baseline?

3.1.2.5 Other Information. This section will be reserved for other AST information not covered in the above sections.

# AST CHECKLIST PROCEDURES

APPENDIX 3

## AST CHECKLIST PROCEDURES

CSA took further steps for better evaluation of procedures by developing a checklist consisting of Appendix 1 and Appendix 2. Each requirement is evaluated step-by-step as opposed to Appendix 2 (Handbook), in which the requirements are evaluated together. The purpose of this checklist is to assure that all requirements are thoroughly evaluated. The checklist is arranged in the following format: first page - general information of the AST; following pages - five (5) columns with first column being item number, second column the title of the AST and its requirements, third column, the paragraph number corresponding to Appendices 1 and 2, and criteria for evaluation. A blank page is also provided for any additional comments. This checklist method was used to demonstrate the AST procedure at the Westinghouse facility in Baltimore, Maryland.

AST TITLE: The name of the tool in its long form.

DATE OF DEVELOPMENT: The date the tool development was (or is to be) completed (month/day/year)

HOST COMPUTER: The hardware identification of the machine on which the tool was developed.

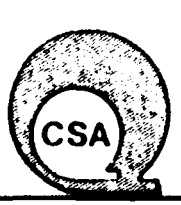
TOOL SUMMARY: Brief information about the tool.

DOCUMENTATION: The written documentation, documentation sources and document reference number.

CONTACT FOR MORE INFORMATION: The contact person(s) to obtain more information about the tool.

NOTE: Questions contained in the various "Criteria for Evaluation" are essentially Yes/No questions. A "YES" response to any question will be supported by documentation for evaluation. A written explanation is also required in the space provided for a "NO" response to any of the questions.





| ITEM | TITLE: GENERAL INFORMATION OF AST (INTRODUCTION) | MONITORING SYSTEMS PARA. NO. 3.1.1.1 | | |
|---|---|---|---|---|
| | REQUIREMENT | CRITERIA FOR EVALUATION | YES | NO |
| 1 | 1. Identify the project(s) to which the AST have or will be applied. | 1a. Has the contractor identified the project(s) to which the AST has or will be applied? | | |
| | 2. List all software tools (in software life-cycle) that interface with the AST. | 2a. Has the AST been evaluated under DoD-STD-2167? | | |
| | | b. Are all software tools listed in software life cycle? | | |
| | 3. Give a brief functional description of the AST. | 3a. Has the contractor provided the AST functional description? | | |
| | | b. Is all documentation for the above information available to government personnel, and furnished upon request? | | |
| | | c. Does the description provide information on input and output and the expected results? | | |

| COMMENTS | PARAGRAPH NO. |
| --- | --- |
| | |

| ITEM | TITLE: GENERAL INFORMATION OF AST (MASTER SCHEDULE PLAN) (Optional) | MONITORING SYSTEMS PARA. NO. 3.1.1.2 | | |
|---|---|---|---|---|
| | REQUIREMENT | CRITERIA FOR EVALUATION | YES | NO |
| 2 | Establish a master schedule indicating the date the AST will become operational. | 2a. Has the contractor provided procedures to describe the master schedule?<br><br>b. Are there provisions for monitoring and tracking the changes that may have effect on the plan?<br><br>c. Has the contractor provided the listings of major milestone schedules?<br><br>d. Has the contractor shown close out of completed tasks? | | |

| COMMENTS | PARAGRAPH NO. |
| --- | --- |
| | |

| ITEM | TITLE: GENERAL INFORMATION OF AST (CONFIGURATION CHANGE AND CONTROL PLAN) | MONITORING SYSTEMS PARA. NO. 3.1.1.3 | | |
|---|---|---|---|---|
| | REQUIREMENT | CRITERIA FOR EVALUATION | YES | NO |
| 3 | Describe how change and control of the AST is maintained. | a. Has the Contractor provided procedures to ensure that changes to baseline specification and documentation are authorized?<br>b. Is the Contractor complying with internal procedures for placement and removal of items from the control facility?<br>c. Has the Contractor followed procedures to identify baseline items and subsequent revision or version of the AST?<br>d. Has the Contractor precluded the control facilities from being used as a repository for unapproved or uncontrolled AST? | | |

| COMMENTS | PARAGRAPH NO. |
| --- | --- |
| | |

THIS PAGE LEFT BLANK INTENTIONALLY

| ITEM | TITLE: GENERAL INFORMATION (PERSONNEL TRAINING PLAN) | MONITORING SYSTEMS PARA. NO. 3.1.1.4 | | |
|---|---|---|---|---|
| | REQUIREMENT | CRITERIA FOR EVALUATION | YES | NO |
| 4 | Describe the overall and individual software tool training plan and the plan to include a training schedule tracking each employee's progress. | a. Has the contractor provided a training plan?<br><br>b. Has a procedure been developed to track the training system?<br><br>c. Are these training plans administered? | | |

| COMMENTS | PARAGRAPH NO. |
| --- | --- |
| | |

| ITEM | TITLE: SPECIFIC SOFTWARE TOOL INFORMATION (AST IDENTIFICATION) | MONITORING SYSTEMS PARA. NO. 3.1.2.1 | | |
|---|---|---|---|---|
| | REQUIREMENT | CRITERIA FOR EVALUATION | YES | NO |
| 5 | Identify the AST and its general characteristics as per (Automated Software Tool Taxonomy attachment 1-1) of Appendix 1. | a. Has the contractor provided information to properly identify the AST?<br><br>b. For the information not provided, are there means to furnish the government the missing information.<br><br>c. Is the AST information accurate and complete?<br><br>d. Is the AST approved to be used on this project?<br><br>e. Is the AST in development stage?<br><br>f. If question 5e is yes, is the development included in the project contract?<br><br>g. Has the Contractor researched existing Government tools to avoid tool duplication.<br><br>h) Are there advantages of this AST compared to the existing toois?<br><br>i) Is the AST a non-deliverable item?<br><br>j) If 5i is yes, are there effective means for providing information to government personnel during and after the development life cycle of software? | | |

| COMMENTS | PARAGRAPH NO. |
| --- | --- |
| | |

| ITEM | TITLE: SPECIFIC SOFTWARE TOOL (SOFTWARE TOOL SCHEDULE) | MONITORING SYSTEMS PARA. NO. 3.1.2.1.2 | | |
|---|---|---|---|---|
| | REQUIREMENT | CRITERIA FOR EVALUATION | YES | NO |
| 6 | Provide a detailed schedule outlining the various activities necessary to make the software tool operational. | a. Has the Contractor provided a milestone schedule for the AST?<br><br>b. Is the information provided in the milestone schedule accurate?<br><br>c. Has the Contractor provided date(s) the AST will become operational? | | |

| ITEM | TITLE: SPECIFIC TOOL INFORMATION (VALIDATION OF SOFTWARE TOOLS & EVALUATION OF DOCUMENTATION) | MONITORING SYSTEMS PARA. NO. 3.1.2.1.3 | | |
|---|---|---|---|---|
| | REQUIREMENT | CRITERIA FOR EVALUATION | YES | NO |
| 7 | 1. Analyze software requirements to determine testability. | 1a. Has the Contractor identified test activities? | | |
| | | b. Are test requirements accurate? | | |
| | | c. Are software tool requirements testable? | | |
| | | d. Have various levels of testing been identified and scheduled as required? | | |
| | 2. Verify that tests are conducted in accordance with approved test plans and procedures. | 2a. Has the contractor provided procedures and documentation for internal testing and evaluation? | | |
| | 3. Review test requirements and criteria for adequacy. | 3a. Do test procedures comply with the test specifications, data item descriptions and other contractual requirements? | | |
| | 4. Review the documentation to verify that it corresponds to the test results. | 4a. Are test results actual findings of test reports? | | |
| | 5. Ensure tests related media and documentation are maintained. | 5a. Are test-related media & documentation maintained to allow repeatability of tests? | | |
| | | b. Is all software and hardware used to develop AST acceptable to the government? | | |

| COMMENTS | PARAGRAPH NO. |
| --- | --- |
| | |

| ITEM | TITLE: SPECIFIC TOOL INFORMATION (CONFIGURATION CHANGE AND CONTROL OF SOFTWARE TOOL DOCUMENTATION) | MONITORING SYSTEMS PARA. NO. 3.1.2.4 | | |
|---|---|---|---|---|
| | REQUIREMENT | CRITERIA FOR EVALUATION | YES | NO |
| 8 | 1. Provide AST baseline date. | 1a. Has the Contractor established the baseline date? | | |
| | 2. Provide AST baseline configuration (if different) from AST identification as specified in Demo No. 5) | 2a. Has the Contractor provided AST baseline configuration? | | |
| | 3. Track changes to AST baseline. | 3a. Have changes been made to the AST baseline? | | |
| | | b. Has the Contractor provided procedures to track changes to baseline? | | |
| | 4. Describe changes to baseline documentation and date(s) of changes. | 4a. Has the Contractor provided a description of the changes made to AST documentation? | | |
| | | b. Have date(s) been provided to baseline changes? | | |

COMMENTS

PARAGRAPH NO.

# EXAMPLE OF TOOL'S CHARACTERISTICS

APPENDIX 4

## OVERVIEW OF AUTOMATED SOFTWARE TOOL

The information below provides general characteristics of an automated software tool. It is presented to help the AFPRO understand what a software tool is and how it works.

There are many ways in which one can view the characteristics of software tools. The approach in this section uses two different vantage points. The first vantage point is a coarse functional view of tools. A very simple classification system that consists of only six categories is used. This view is followed by the second vantage point which is a much more detailed perspective that is based on the taxonomy (characteristics) of tool features. Each feature is described and defined.

### Tool Classification

The tools in the STI Database are not extensively classified according to traditional schemes because of their limitation in describing current technology. The number of categories in the traditional classification scheme was reduced to six. The following table lists the six categories and the number of tools identified in each category:

### Tool Class

Software Management, Control and Maintenance
Software Modeling and Simulation
Requirements and/or Design Specification and Analysis
Source Program Analysis and Testing
Program Construction and Generation
Software Support System/Programming Environments

Since the above classes are not mutually exclusive, this categorization provides only a broad overview of the types of tools currently available. If another classification exists, the classification and a definition of the classification should be presented.

### Tool Features

To provide a more useful way of identifying tools of interest, each of the tools in the STI Database is classified according to the taxonomy of

## Tool Features

To provide a more useful way of identifying tools of interest, each of the tools in the STI Database is classified according to the taxonomy of (general characteristics) of tool features. The features taxonomy, displayed in Figure 1, graphically illustrates the hierarchical relationship of tool features. The INPUT, FUNCTION, and OUTPUT categories are described below.

### Input

Tool input features are based on the forms of input which can be provided to a tool. These features fall into two classes, one based on what the tool operates on (Subject), and the other based on how the tool operates (Control Input). The difference between these classes is clarified further in the following paragraphs.

#### Subject.

The subject is usually the main input to a tool. It is the input which is subjected to the main functions performed by a tool. The four types of tool subjects are: code, very high level language (VHLL), data, and text. Although the difference between these types is somewhat arbitrary, the taxonomy has very specific definitions for each.

(1) Code Input - accepts a program written in a high level, assembly or object language. Code is the language form in which most programming solutions are expressed.

(2) VHLL Input - accepts a program written in a very high level language that is typically not in an executable form. Tools with this feature may define programs, track program requirements throughout their development, or synthesize programs through use of some non-procedural VHLL. Typical VHLLs are:

- Design Specification Language
- Requirements Specification Language
- Program Specification Language
- System Specification Language

- Algebraic Specification Language
- Model Specification Language
- Description Language
- Structured Language
- Requirements Language
- Design Language
- Specification Language

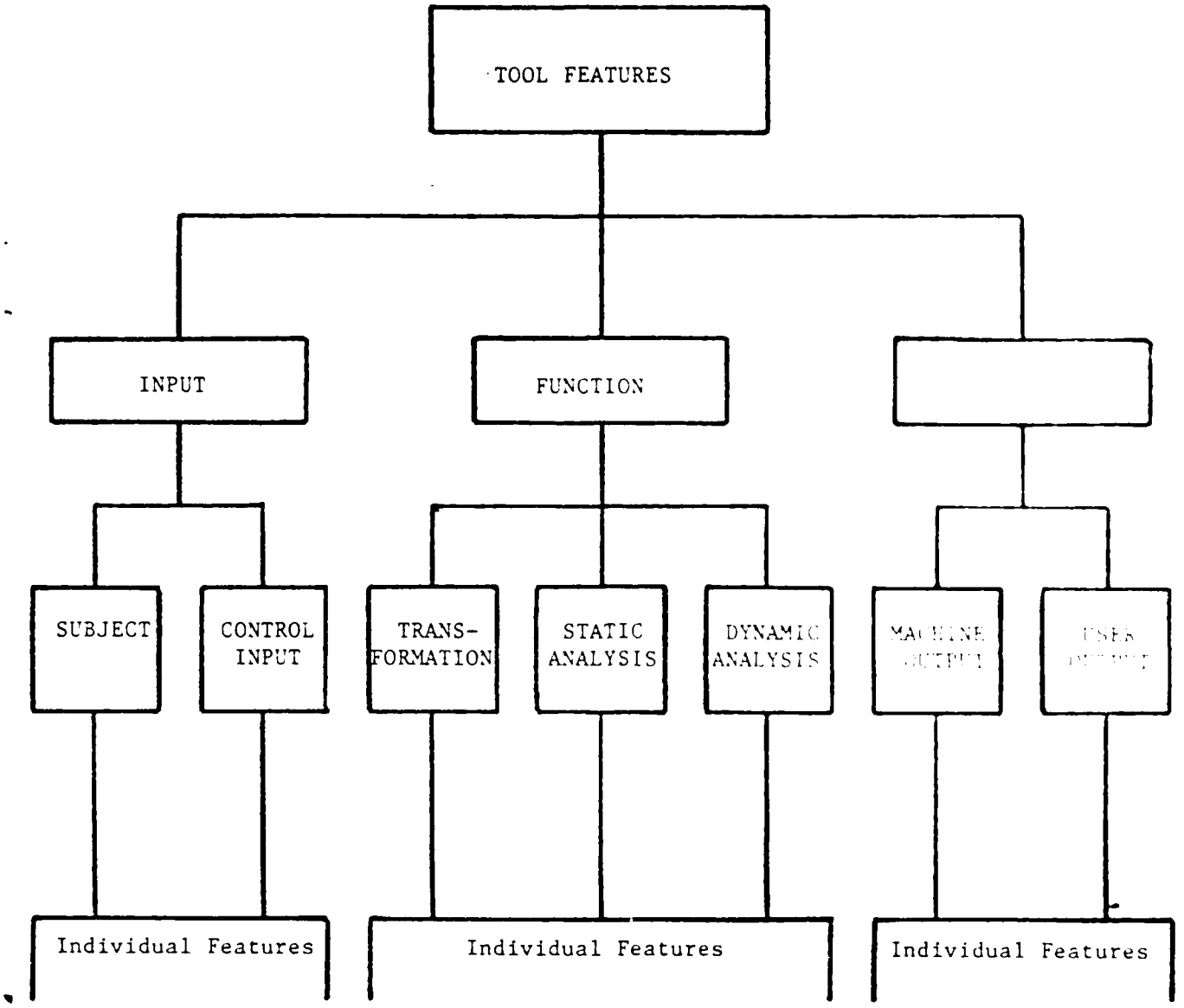



FIGURE 1: TAXONOMY OF TOOL FEATURES

(3) Data Input - accepts a string of characters to which meaning is or might be assigned. The input (e.g. raw data) is not in an easily interpreted, natural language form.

(4) Text Input - accepts statements in a natural language form. Certain types of tools are designed to operate on text only (e.g., text editors, document preparation systems) and require no other input except directive or commands.

Control Input.

Tools that have control input features accept statements or data that specify the type of operations and any detail associated with the operations. They describe any separable commands that are entered as part of the input stream (HOUG82B). The following describe the two control inputs features:

(1) Commands - accepts character strings which consist primarily of procedural operators, each capable of invoking a system function to be executed. A directive invoking a series of diagnostic commands (i.e., TRACE, DUMP, etc.) at selected break-points is an example. A tool that performs a single function will not have this feature but will most likely have the next (HOUG82B).

(2) Parameters - accepts character strings which consist of identifiers that further qualify the operations to be performed by a tools. Parameters are usually entered as a result of a prompt from a tool or may be embedded in the tool input. An interactive trace routine that prompts for break-points in an example of a tools with parametric input (HOUG82B).

The following figures provides an enlarged illustration of the INPUT category in Figure 1.

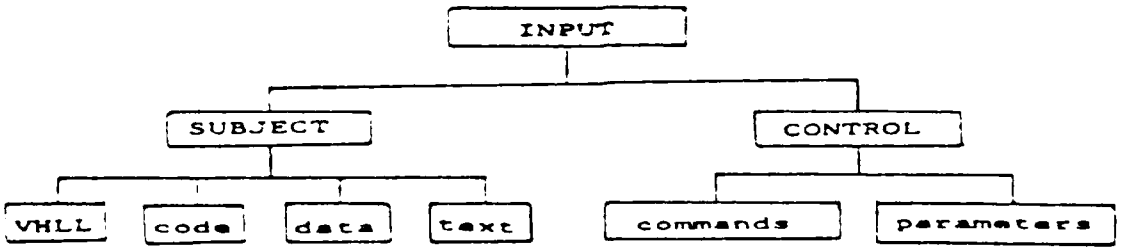



FIGURE 2

Function

The features for this class describe the processing functions performed by a tool and fall into three classes: Transformation, Static Analysis and Dynamic Analysis.

Transformation.

Transformation features describe how the subject is manipulated to accommodate the user's needs. They describe what transformations take place as the input to the tool is being processed. The following list includes the features that are classified as transformation:

(1) Formatting is arranging a program according to predefined or user defined conventions. A tool that "cleans up" a program by making all statement numbers sequential, alphabetizing variable declarations, indenting statements, and making other standardizing changes has this feature.

(2) Translation is coverting from one language form to another. The following lists includes a representative sample of features that are classified as Translation:

- structured preprocessing
- ccmplication
- macro expansion
- conversion

(3) Instrumentation is adding sensors and counters to a program for the purpose of collection information useful for dynamic analysis. Most code analyzers instrument the source code at strategic points in the program to collect execution statistics required for coverage analysis and tuning. See Dynamic Analysis features.

(4) Editing is modifying the content of the subject by inserting, deleting, or moving characters, numbers, or data. Of course, there are a very large number of tools that have this feature that are not included in the STI Database. Many of the ones that are included provide editing to enhance the capability of the user environment.

(5) Synthesis is generating an application or program from a specification or from an intermediate language. Tools that have this feature include application generators, program generators, compiler compilers and preprocessor generators.

(6) Restructuring is reconstructing and arranging the subject in a new form according to well-defined rules. A tool that generates structured code from unstructured code is an example of a tools with this feature.

(7) Optimization is modifying a program to improve performance, e.g. to make it execute faster or to make it use fewer resources.

Static Analysis

Static Analysis features describe operations on the subject without regard to its executability. They describe the manner in which the subject is analyzed. All management-related operations are classified at Static Analysis. The following list includes the features that are classified as Static Analysis:

(1) Management is aiding the management or control of software development. The following includes a representative sample of the features that are classified as Management:

- configuration
- global variable
- project management
- database management
- change control
- test data management
- files management
- library management
- version control
- documentation management
- performance management
- capacity planning
- management planning

(2) Cross Reference is referencing entities to other entities by logical means.

(3) Scanning is examining an entity sequentially to identify key areas of structure.

(4) Auditing is conducting an examination to determine whether or not predefined rules have been followed.

(5) Data Flow Analysis is graphical analysis of the sequential patterns of of definitions and references of data.

(6) Consistency Checking is determining whether or not each entity is internally consistent in that it contains uniform notation and terminology and is consistent with its specification.

(7) Statistical Analysis is performing statistical data collection and analysis.

(8) Error Checking is determining discrepancies, their importance and/or their cause.

(9) Structure Checking is detecting structural flaws within a program (e.g. improper loop nestings, unreferenced labels, unreachable statements with no successors).

(10) Comparison is determining and assessing differences between two or more items.

(11) Completeness Checking is assessing whether or not an entity has all its parts present and if its parts are fully developed.

(12) Complexity Measurement is determing how complicated an entity (e.g., routine, program, system, etc.) is by evaluating some number of associated characteristics. For example, the following characteristics can impact complexity: instruction mix, data references, structure/control flow of interactions/interconnections, size, and number of computations.

(13) Tracking is tracking the development of an entity through the software life cycle.

(14) Interface Analysis is checking the interfaces between program elements for consistency and adherence to predefined rules and/or axioms.

(15) I/O Specification Analysis is analyzing the input and output specification in a program, usually for the purpose of generating input data.

(16) Type Analysis is evaluating whether or not the domain of values attributed to an entity are properly and consistently defined.

(17) Cost Estimation is assessing the behavior of the variables which impact life cycle cost.

(18) Units Analysis is determining whether or not the units or physical dimensions attributed to an entity are properly defined and consistently used.

(19) Scheduling is assessing the software development schedule and its impact on the software life cycle.

Dynamic Analysis

Dynamic Analysis features specify operations that are determined during or after execution takes place. Dynamic Analysis features differ from those classified as static by virtue of requiring some form of symbolic or machine execution. They describe the techniques used by the tool to derive meaningful information about a program's execution behavior. The following lists includes the features that are classified as Dynamic Analysis:

(1) Coverage Analysis is determining and assessing measures associated with the invocation of program structural elements to determine the adequacy of a test run. Coverage Analysis is valuable when the user is attempting to execute each statement, branch, path or interactive structure (i.e., Do loops in FORTRAN) in a program.

(2) Tracing is tracing the historical record of execution of a program. The following list includes a representative sample of the features that are classified as Tracing:

- path flow tracing
- break-point control
- logic flow tracing
- data flow tracing

(3) Tuning is determining what parts of a program are being executed the most.

(4) Simulation is representing certain features of the behavior of a physical or abstract system by means of operations performed by a computer.

(6) Resource Utilization is analysis of resource utilization associated with system hardware or software.

(7) Symbolic Execution is reconstructing logic and computations along a program path by executing the path with symbolic, rather than actual values of data.

(8) Assertion Checking is checking of user-embedded statements that assert relationships between elements of a program. An assertion is a logical expression that specifies a condition or relation among the program variables. Checking may be performed with symbolic or run-time data.

(9) Regression Testing is rerunning test cases which a program has previously executed correctly in order to detect errors spawned by changes or corrections made during software development and maintenance.

(10) Constraint Evaluation is generating and/or solving path input or output constraints for determing test input or for proving programs correct.

The following figure provides an enlarged illustration of the FUNCTION category in Figure 1.

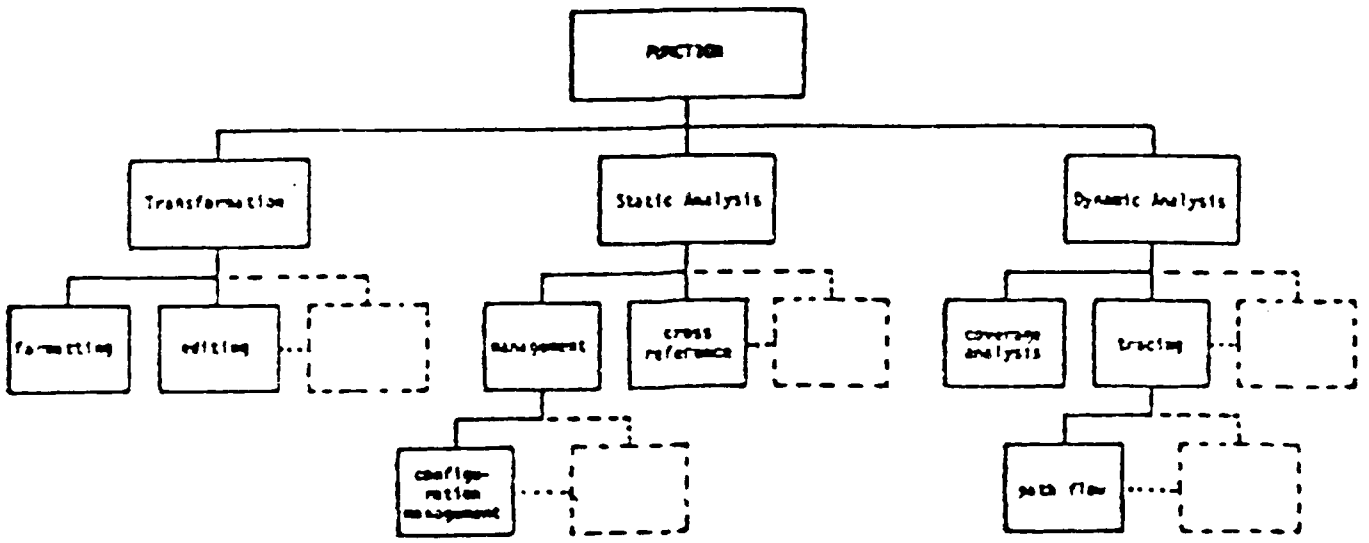



FIGURE 3

Output

Output features provide links from the tool to both the human user and the target machine (where applicable). They describe the types and forms of outputs that are produced by a tool.

User Output. User output features describe the types of information that are returned from the tool to the human user and the forms in which these outputs are presented. The following includes the features which fall into this class:

(1) Listings are output that lists source programs or data and that may be annotated.

(2) Tables are output that is arranged in parallel columns to exhibit a set of facts or relations in a definitive, compact and comprehensive form.

(3) Diagnostics are output that simply indicate what software discrepancies have occurred.

(4) Graphics are a presentation with symbols indicating operations, flow, etc. The following list includes a representative sample of the output features that are considered Graphics:

- flow charts
- hierarchical tree
- design charts
- activity diagram
- charts
- HIPO charts
- line graphs
- bar charts
- control map
- histograms
- milestone charts
- activity diagrams
- structure charts

(5) User-Oriented Text is output that is in a natural language form. User-Oriented Text is further extended into the following areas:

- documentation
- reports

Machine Output.

Machine Output features handle the interface from the tool to a non-human user. The machine output can be directed to a target machine or to another tool for further processing. Machine Output features describe what the receiving tool or machine expects as output. The following list includes the features that fall into this class:

(1) Source Code is a program written in a procedural language that must be input to a translation process before execution can take place.

(2) Data is a set of representations of characters or numeric quantities which meaning has been assigned.

(3) Object Code is a program expressed in machine language which is normally an output of a given translation process.

(4) Intermediate Code is a code that is between source code and machine code.

(5) VHLL is a program written in a very high level language.

(6) Prompts are a series of procedural operators that are used to interactively inform the system in which to tool operates that it is ready for the next input.

The following figure provides an enlarged illustration of OUTPUT from Figure 1.

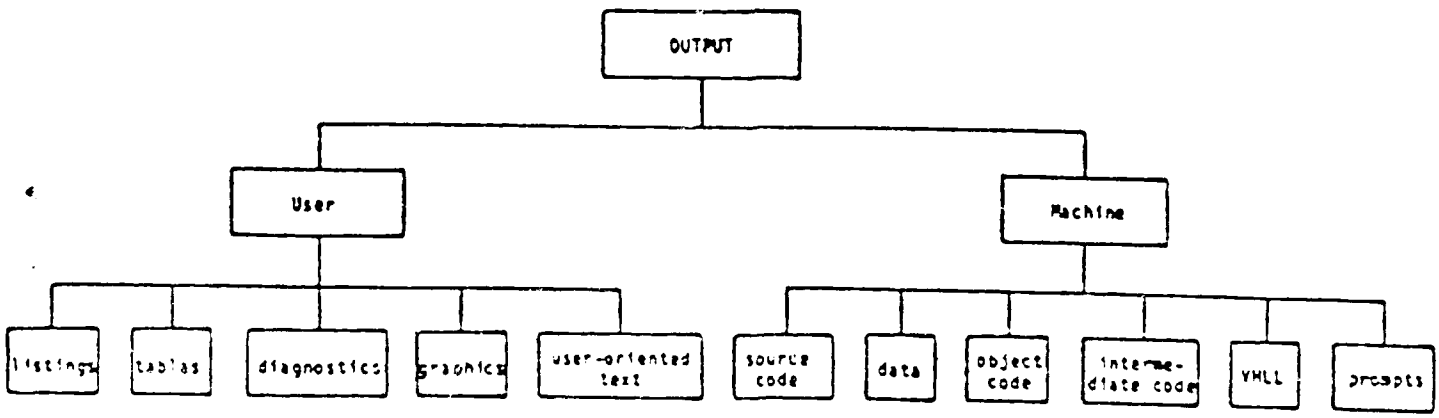



FIGURE 4

END

4-87

DTIC